PUSL3120
FULL-STACK DEVELOPMENT

20 CREDIT MODULE

ASSESSMENT: 100% Coursework **W1: 40% Set Exercises**
W2: 60% Project Report

MODULE LEADER: Dr Mark Dixon

MODULE AIMS

- This module explores the production of dynamic web applications with a particular focus on the web environment.
- Key elements such as object oriented and event-based scripting, asynchronous client-server communication and distributed content representation are explored through practical production.
- The production of working systems use frameworks such as HTML, CSS, JavaScript/JQuery and Node.js.

ASSESSED LEARNING OUTCOMES (ALO):

1. Apply principles of object oriented and event-based scripting as well as synchronous and asynchronous client-server communication.
2. Demonstrate an understanding of how application content is represented and communicated across the web and how this affects the user experience.
3. Design, implement and evaluate/test dynamic web-based applications.

Overview

This document contains all the necessary information pertaining to the assessment of *PUSL3120 Full-Stack Development*. The module is assessed via **100% coursework**, across two elements: *40% Set Exercises (Individual)* and *60% Project Report (Group)*.

The sections that follow will detail the assessment tasks that are to be undertaken. The submission and expected feedback dates are presented in Table 1. All assessments are to be submitted electronically via the respective DLE module pages before the stated deadlines.

	Submission Deadline	Feedback
Group Project Report (60%)	**Mon 8th Jan 2024**	Mon 5th Feb 2024
Individual Set Exercises (40%)	**Tue 9th Jan 2024**	After submission

Table 1: Assessment Deadlines

All assessments will be introduced in class to provide further clarity over what is expected and how you can access support and formative feedback prior to submission. Whilst the assessment information is provided at the start of the module, you are not expected to start this immediately (as you will often not have sufficient understanding of the topic). The module leader will provide guidance in this respect.

As with any software development project, the details of this brief may change slightly over time. You must check for changes and announcements (both email and in teaching sessions) regularly.

This document was last updated: Friday, 01 December 2023

Assessment 1: Set Exercises (Web Development)

Task:

This coursework contributes **40%** of the overall module mark for COMP3006 and is an **individual assignment**. It will be administered via the DLE quiz facility and as such will be run entirely online. Students will be given a 3 hour period in which to submit. They will be allowed up to 3 attempts within that period and the final mark will be an average of all attempts taken. The mark should be available just after submission.

Note: This is separate from the lab exercises that are handed out each week (which are not assessed).

Assessment 2: Full-Stack Project

Task:

This assignment contributes **60%** of the overall module mark for COMP3006 and is an **individual assignment**. The **final submission** must be submitted to the DLE by the specified submission dates.

This is a negotiated project in which you must define the specific content you will produce and the process you will follow, within the guidelines given below. Your project **must satisfy** the following requirements:

- You must produce a working full-stack system, comprising
  - a client constructed using dynamic web technologies (***HTML***, ***CSS*** and ***JavaScript***)
  - a server using Node.js (no other server is permitted), and
  - a database (either **MongoDB or PouchDB**) to store information and pass it to the client through an API. Other types of database are not acceptable for this module.
- ***WebSockets*** should be used to enable communication between two clients.
- The application must use ***JavaScript*** (including typescript) as the main development language both client-side and server-side (no other language is acceptable).
- The system must be ***interactive***, i.e., the user should be able to affect its behaviour by interacting with it using a suitable input device (such as keyboard and mouse).
- The system must run on multiple computers, i.e., it must be ***distributed*** (i.e. stand-alone applications are not permitted).
- Include CRUD functionality for 1 entity per group member (at least 4 entities – minimum group size is 4 students, maximum group size is 6 students).
- Include appropriate security considerations (such as a login system)
- Be selected from ***one*** of the following topics and include appropriate functionality:
  - School Learning Platform
  - Project Management System
  - eCommerce System

- Hotel Room Booking System
- Library System
- Cinema Booking System
- GP / Hospital / Dental Appointment Booking System
- Gym Workout Record System
- Restaurant / Takeaway Ordering System
- Travel (Bus / Train / Flight) Booking System

The final product should reflect an effort of 80+ hours of dedicated work.

You must document the system, including its design and details of the implementation process you have followed, and provide a description of your DevOps pipeline. This should include your code repository, continuous integration/deployment setup, unit tests, integration tests, behaviour tests, code analyses, usage metrics and usage analyses.

COURSEWORK DELIVERABLES

There are three main deliverables for this assignment.

D1 – Report

For this deliverable you must submit **a report file (in PDF format)** containing a written report describing your project. The report must include (on the first page):

- A link to your code on GitHub
  (no other source will be marked, you must ensure it can be accessed).
- A link to your video on YouTube
  (no other source will be marked, you must ensure it can be accessed).

The report must be a document of no more than 2,000 words. Please use screen shots and sketches to illustrate the functionality and UML diagrams (or appropriate alternatives) to illustrate the software architecture / component design.
The report should explain:

- Requirements (ca. 400 words with additional documents in appendices)
  - Who is the application aimed at?
  - What features were included and why?
- Design (ca. 500 words with UML diagrams)
  - What is the system architecture (clients/servers/peers)?
  - How do the components interact?
  - How are the data and code structured?
  - Why are these structures appropriate?
- Testing (ca. 400 words)
  - What automated testing have you performed?

    - What usability testing have you performed?
- DevOps pipeline (ca. 400 words)
    - Describe your development environment.
    - Describe your continuous integration pipeline and how you used it.
- Personal reflection (ca. 300 words)
    - What worked/didn't work well (techniques and technologies you used)?
    - What lessons would you take from this project into your next project.

Please check your submitted files are correct by downloading them again and checking that they work. **You should receive a confirmation receipt by email when your work has been properly submitted** – if you do not receive this email then your work has not been submitted.

D2 – Source Code

The code you have written (both the software and the tests). This must be shared on GitHub and a link included in your report (D1). Use your real names for your GitHub account, not nick names, and share it with the account *md-supervisor*. This should contain all of the code you have written yourself and should (including copies of the folders and files needed to run your application). Failure to submit a working link within your report is likely to lead to a substantially reduced (failing) mark

D3 – Video

A video of 10 minutes (***only the first 10 minutes will be marked***) in which the group presents a demonstration of the main functionality working, and a brief demo of the supporting processes (tests running and DevOps process). A link must be included in your report (D1) and the video must be accessible.
Each team member should present at least the functionality for the entity(s) they are responsible for. There must be a single video (not separate links for each group member – if multiple videos are included only the first will be marked).
The video **must be uploaded as an unlisted YouTube video**. The video will be used to assess the implementation of the software, and failure to submit a working link within your report is likely to lead to a substantially reduced (failing) mark.

These deliverables must be submitted to the DLE by the submission deadline.

Team Formation/Allocation
Students have the opportunity to form their own teams by mutual agreement up until 17:00 on Fri 8th December 2023. Students without a team after this date will be assigned by the module leader / local tutor.

Common Problems
Based on past experiences running the module, these are the following cases of some common contingencies encountered and how to manage them:

1) **Project changes** – the team are able to change the project topic and / or the proposed functionality and scope as the project progresses if deemed appropriate. However, they are responsible for those changes.
2) **Changing team** – no team member changes are allowed unless under cases:
    a) under item 6 below or
    b) addition of new members to the team under the module leader's discretion. Case b) may be due to one team member left due to other members dropping the module and needs to be reallocated, or new enrolments.
3) **Non-engaging team member** – the definition of a non-engaged team member is one who is uncontactable for a maximum of 2 weeks. If this happens, speak to the local tutor for necessary action (or the module leader if they are visiting NSBM). All procedures during this period must be fully documented.
4) **Mental/Physical Health Issues** - Members having mental/physical health issues are expected to notify their teammates about periods of absence, which should be recorded under minutes. Teams should plan work around these contingencies. Disability services support letter is required for such absences.
5) **Dispute in contribution** – all marks are to be divided equally among team members. In case this is not agreed upon by team members, initially speak to the local tutor. You may then choose to send a marks reallocation email (with form attached and all members cc-ed) on how to scale the product marks. This needs to be made to the module leader before the submission deadline, with the relevant scale and rationale articulated clearly. Settle the dispute internally before emailing this request, as one team can only send this once (only the first email will be considered, subsequent requests will be ignored). If help is required to remediate, the module leader/tutor will call a meeting prior to the request being sent. The form will be made available on the DLE later in the semester.
6) **Disputes in project design and direction** – these disputes are to be resolved internally within the team under consensus but be professional during the process.
7) **Professional conduct** – team members who behave unprofessionally and do not show good team spirit in working together may be penalised if there are multiple complaints from various team members. It is in everyone's best interest to work well together and settle disputes amicably. Learn to compromise and practice polite communication.

The handling of all disputes will be done in consensus by module leader and tutor, and in certain cases under advisement by other staff (such as Stage Tutor and Programme Manager). Decisions made will be final.

Assessment Criteria:

Your work will be assessed according to the rubric found in Table 2. Your mark for this piece of coursework will be based on an aggregation of the marks for each category. Marks will be awarded based on **the video, code and the report**.

Category	Fail (< 40%)	>= 40%	>= 50%	>= 60%	>= 70%
Analysis (10%)	Insufficient problem analysis. Functional requirements are not described in sufficient detail.	There are some vague functional requirements based on a basic outline of the problem.	Reasonably detailed functional requirements but based on weak analysis.	Functional requirements are well specified with some evidence of strong analysis.	Functional requirements are complete and are well motivated based on strong analysis.
Design (10%)	Little or no design work. It is not clear how the system will be built, or what it's major components are.	Some overview of the design but no detail. The architecture is not defined. Little or no use of diagrams to show the structure and operation of specific sections of the system.	Some of the system's design is specified in detail but other areas are weak. Some diagrams are included.	Design is mostly complete, further detail would enhance some areas. Most functionality and design specified with diagrams.	System design complete and comprehensively described with the use of appropriate diagrams.
Testing (20%)	Little or no testing of any kind.	Some testing – either code (unit or integration) or usability testing has been completed, but large parts of the system are untested.	A reasonable level of testing – either code (unit or integration) or usability testing has been completed. A good amount of the system is tested in some way.	Reasonably complete code (unit or integration) and usability testing. A good amount of the system is tested with both approaches.	Comprehensive code testing with both unit and integration tests. A thorough usability study has been performed and feedback has been taken on board. Other types of testing (e.g., load) have been used.
CI/CD (10%)	No attempt at an automated CI pipeline.	No automated CI pipeline. Some use of version control.	Good use of version control but no automated CI pipeline.	Version control has been used effectively. Some indication of test-driven development. CI pipeline has been attempted.	Version control is used well. CI pipeline is well organised and has been demonstrated to work.
Evaluation (10%)	Little or no evaluation.	Some evaluation, but lacking both breadth and detail.	Some detailed evaluation of limited range of aspects of project, but other areas weak.	Good evaluation with broad coverage and detail.	Comprehensive evaluation of system (both detailed and broad).

Software (40%)	Very little software has been constructed, and what there is does not work.	Most of the functionality has not been implemented. Major errors at runtime. WebSockets and/or a database have been attempted but incomplete. Much of the system is affected by bugs.	Some requirements implemented but major omissions. Software contains major runtime errors. WebSockets and database used, one of them weakly The software has some bugs affecting the main functionality.	Largely complete implementation of most requirements. Some runtime errors. WebSockets and a database are used, both work correctly. There are some apparent bugs but they do not impede the main functionality.	Implementation of all requirements is complete. Very few apparent bugs. The approach taken to implementing the database and WebSockets is nearing or at a commercial standard.

Table 2: Feedback Template for Assessment 1

General Guidance

Extenuating Circumstances

There may be a time during this module where you experience a serious situation which has a significant impact on your ability to complete the assessments. The definition of these can be found in the University Policy on Extenuating Circumstances here: https://www.plymouth.ac.uk/student-life/your-studies/essential-information/exams/exam-rules-and-regulations/extenuating-circumstances

For this module the way the CW1/W1 Set exercises are managed means you will have ample opportunity to overcome any minor setbacks during the time of the module and still be able to deliver your best work.

Plagiarism

All of your work must be of your own. You must use references for your sources, however you acquire them. Where you wish to use quotations, these must be a very minor part of your overall work.

To copy another person's work is viewed as plagiarism and is not allowed. Any issues of plagiarism and any form of academic dishonesty are treated very seriously. All your work must be your own and other sources must be identified as being theirs, not yours. The copying of another persons' work could result in a penalty being invoked.

Further information on plagiarism policy can be found here:

Plagiarism: https://www.plymouth.ac.uk/student-life/your-studies/essentialinformation/regulations/plagiarism

Examination Offences: https://www.plymouth.ac.uk/student-life/your-studies/essentialinformation/exams/exam-rules-and-regulations/examination-offences

Turnitin (http://www.turnitinuk.com/) is an Internet-based 'originality checking tool' which allows documents to be compared with content on the Internet, in journals and in an archive of previously submitted works. It can help to detect unintentional or deliberate plagiarism.

It is a formative tool that makes it easy for students to review their citations and referencing as an aid to learning good academic practice. Turnitin produces an 'originality report' to help guide you. To learn more about Turnitin go to:
https://help.turnitin.com/feedback-studio/blackboard/basic/student/student-category.htm